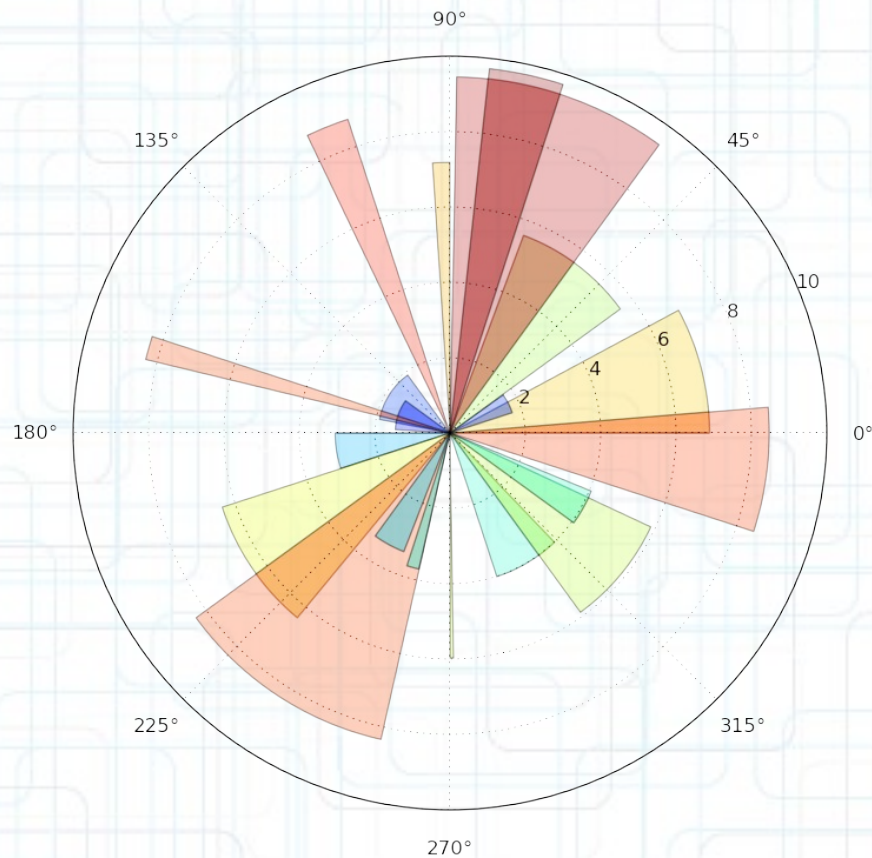


# Matplotlib



- Short Course in Python
- Lecturer: Sławomir
  - Today: Matplotlib
  - Presentation: Saeed
  - Date: 3rd April 2013

# John Hunter - RIP

- the creator of matplotlib
- founding board member of NumFOCUS
- diagnosed with cancer in late July 2012
- passed away on August 28th
- donate to the John Hunter Memorial Fund if you wish
  - For education of Clara, Ava, and Rahel





# Too Big? Where to start?

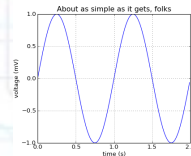
- Overview:  
<http://matplotlib.org/contents.html>
  - User's Guide:  
<http://matplotlib.org/users/>
  - FAQ (How to):  
[http://matplotlib.org/faq/howto\\_faq.html](http://matplotlib.org/faq/howto_faq.html)
  - ...
- I would search for target:
  - The Gallery:  
<http://matplotlib.org/gallery.html>
  - Screen shots:  
<http://matplotlib.org/users/screenshots.html>
  - Examples:  
<http://matplotlib.org/examples/index.html>

# Gallery

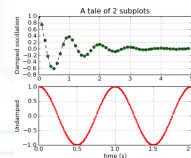
- Categories:
  - Api
  - pylab\_examples
  - mplot3d
  - widgets
  - axes\_grid
- What is there?
  - Lots of figures, each representing a technique/function of plotting
  - Clicking on each figure, you get the source code for that.

# Screen Shots

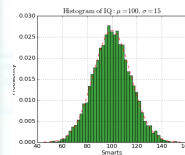
- Simple Plot



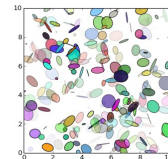
- Subplot



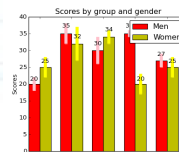
- Histograms



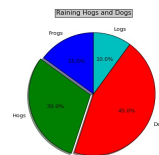
- Ellipses



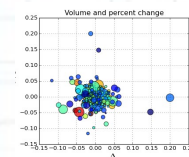
- Bar charts



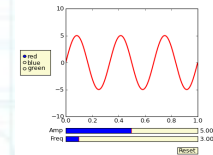
- Pie charts



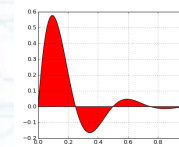
- Scatter



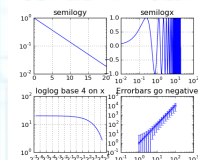
- Slider



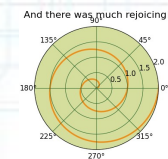
- Fill



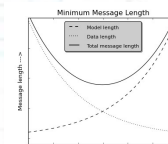
- Log plots



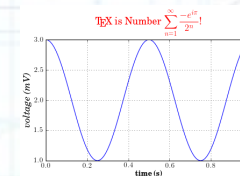
- Polar plots



- Legends



- TeX





# Screen Shots

- Simple Plot
- Subplot demo
- Histograms
- Path demo
- mplot3d
- Ellipses
- Bar charts
- Pie charts
- Table demo
- Scatter demo
- Slider demo
- Fill demo
- Date demo
- Financial charts
- Basemap demo
- Log plots
- Polar plots
- Legends
- Mathtext\_examples
- Native TeX rendering
- EEG demo

# Example#1 Simple Line Plot

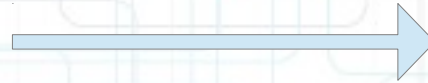
```
from pylab import *
```

```
x = arange(0.0, 2.0, 0.01)
```

```
y = sin(2*pi*x)
```

```
plot(x, y, linewidth=1.0)
```

```
show()
```



Numpy

+

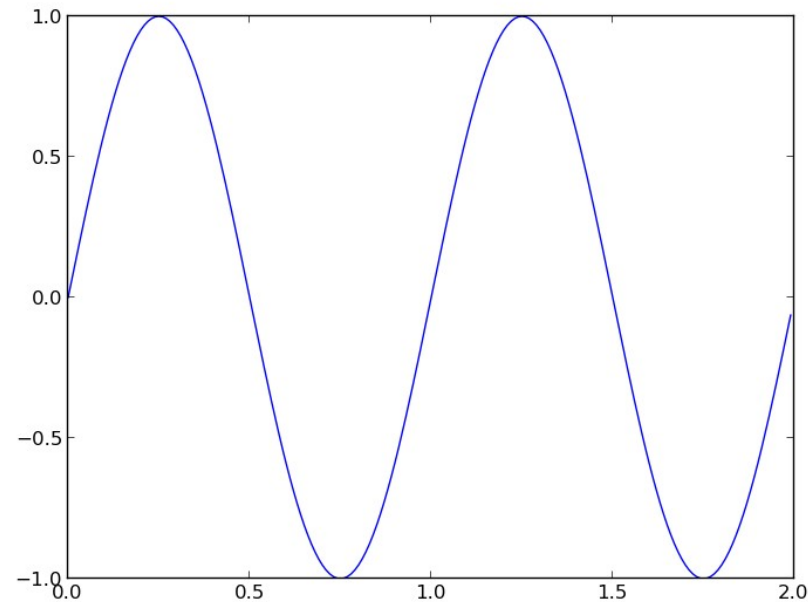
scipy

+

ipython

+

matplotlib



# Example#1 Simple Line Plot

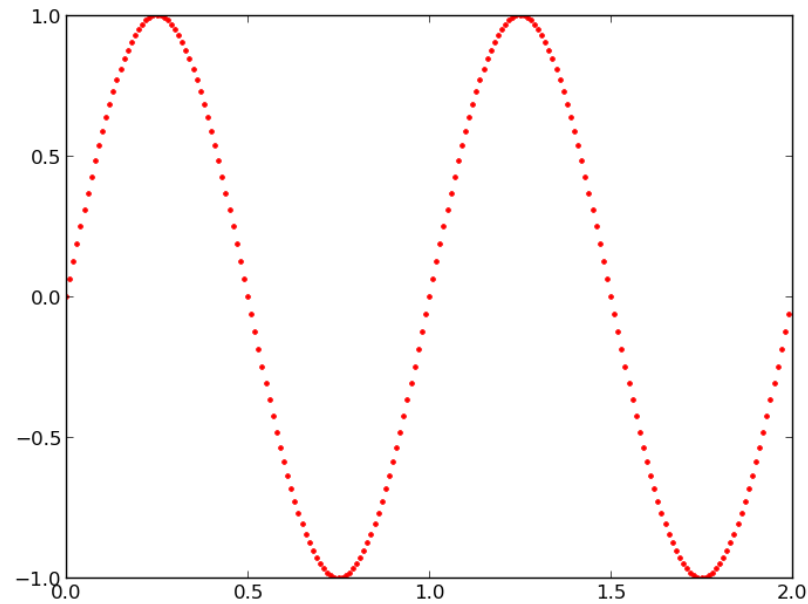
```
from pylab import *
```

```
x = arange(0.0, 2.0, 0.01)
```

```
y = sin(2*pi*x)
```

```
plot(x, y, 'r.')
```

```
show()
```





# Example#1 Simple Line Plot

```
from pylab import *
```

```
x = arange(0.0, 2.0, 0.01)
```

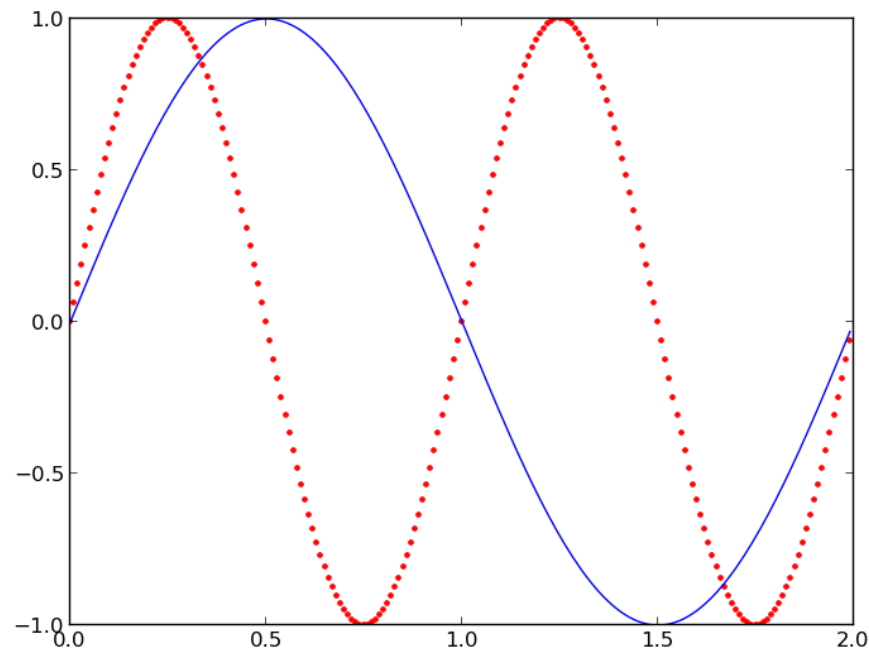
```
y = sin(2*pi*x)
```

```
plot(x, y, 'r.')
```

```
y = sin(pi*x)
```

```
plot(x, y, 'b-')
```

```
show()
```



# Example#1 Simple Line Plot

```
from pylab import *
```

```
x = arange(0.0, 2.0, 0.01)
```

```
y = sin(2*pi*x)
```

```
plot(x, y, 'r.')
```

```
y = sin(pi*x)
```

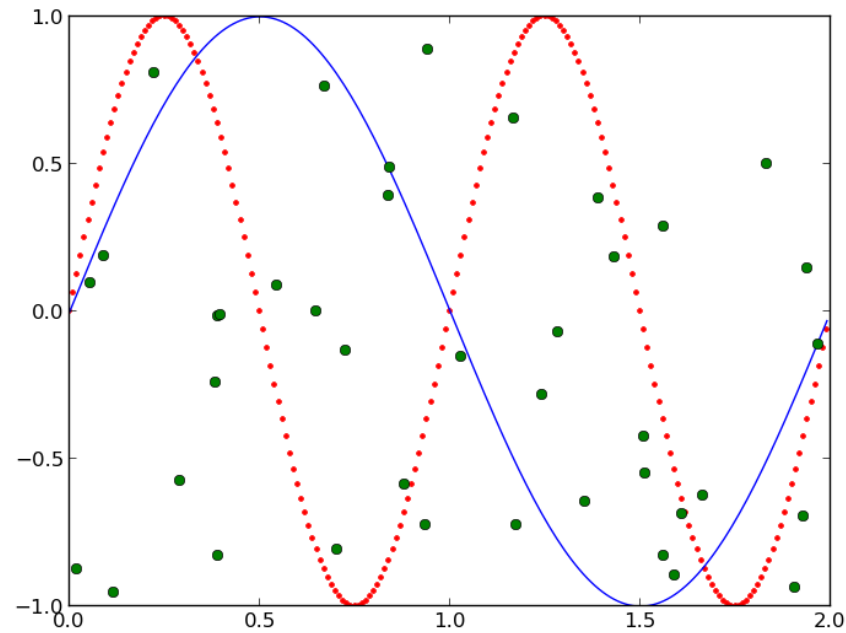
```
plot(x, y, 'b-')
```

```
X = 2 * rand(40)
```

```
Y = 2 * rand(40) - 1
```

```
plot(x, y, 'go')
```

```
show()
```



# Example#1 Simple Plot

```
from pylab import *
```

```
x = arange(0.0, 2.0, 0.01)
```

```
y = sin(2*pi*x)
```

```
plot(x, y, linewidth=1.0)
```

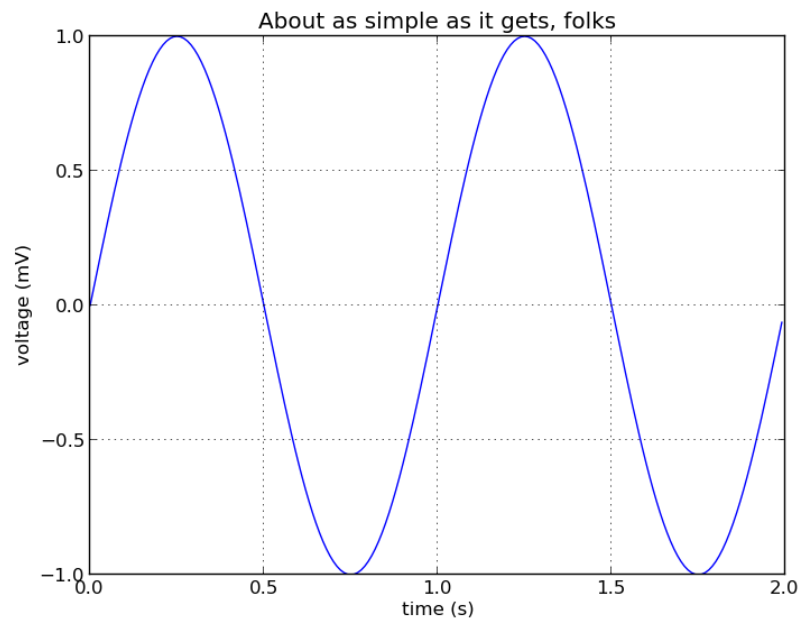
```
xlabel('time (s)')
```

```
ylabel('voltage (mV)')
```

```
title('About as simple as it gets, folks')
```

```
grid(True)
```

```
show()
```





# Example#2 Subplot

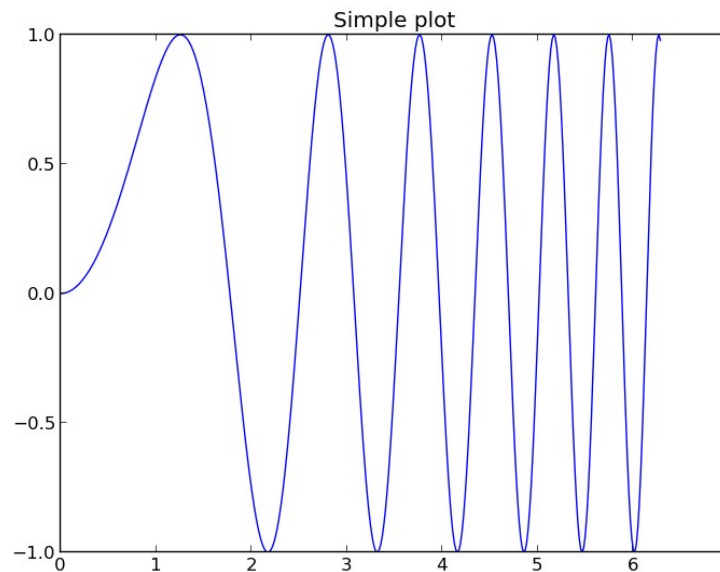
```
import matplotlib.pyplot as plt  
import numpy as np
```

```
x = np.linspace(0, 2 * np.pi, 400)  
y1 = np.sin(x ** 2)
```

```
f, ax = plt.subplots()  
ax.plot(x, y1)  
ax.set_title('Simple plot')
```

```
plt.show()
```

- SINGLE PLOT



# Example#2 Subplot

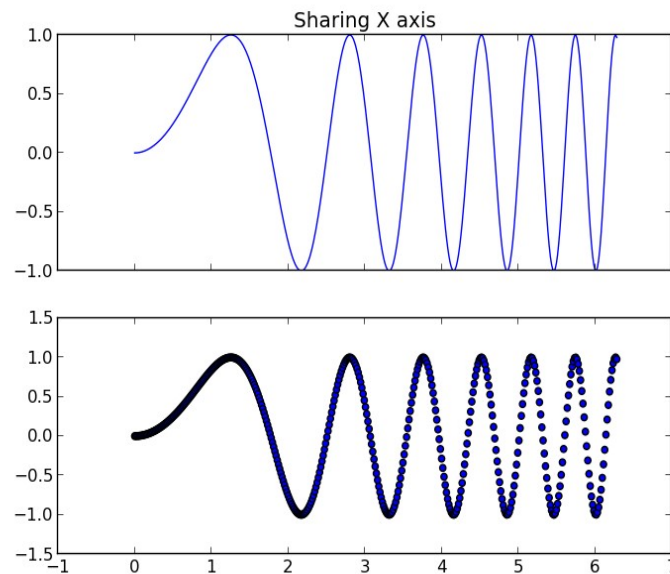
```
import matplotlib.pyplot as plt  
import numpy as np
```

```
x = np.linspace(0, 2 * np.pi, 400)  
y = np.sin(x ** 2)
```

```
f, axarr = plt.subplots(2, sharex=True)  
axarr[0].plot(x, y)  
axarr[0].set_title('Sharing X axis')  
axarr[1].scatter(x, y)
```

```
plt.show()
```

- 2 SUBPLOT
- Axes of subplots **in an array**
- Sharing **x-axis**





# Example#2 Subplot

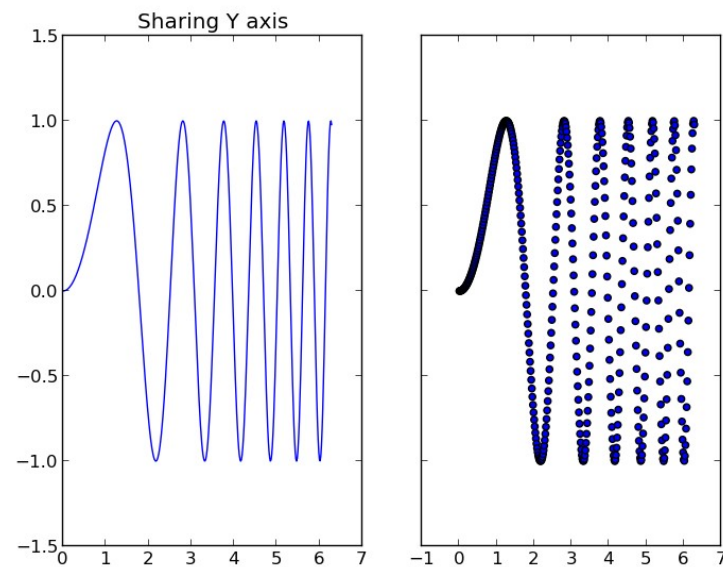
```
import matplotlib.pyplot as plt  
import numpy as np
```

```
x = np.linspace(0, 2 * np.pi, 400)  
y = np.sin(x ** 2)
```

```
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)  
ax1.plot(x, y)  
ax1.set_title('Sharing Y axis')  
ax2.scatter(x, y)
```

```
plt.show()
```

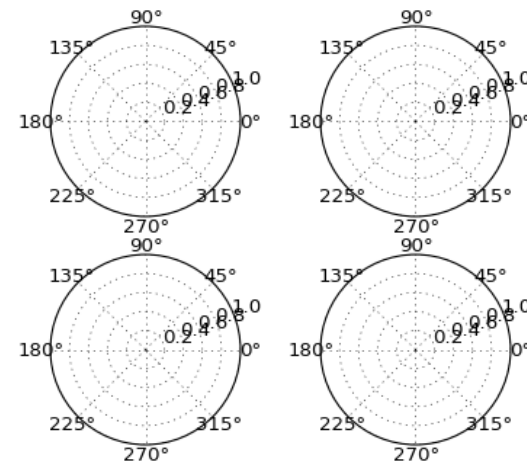
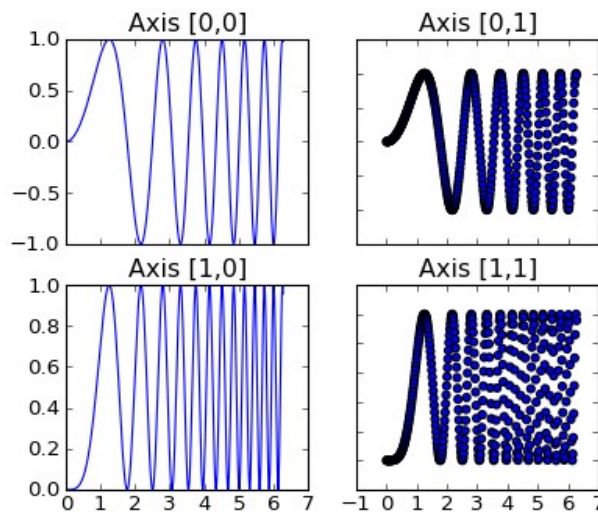
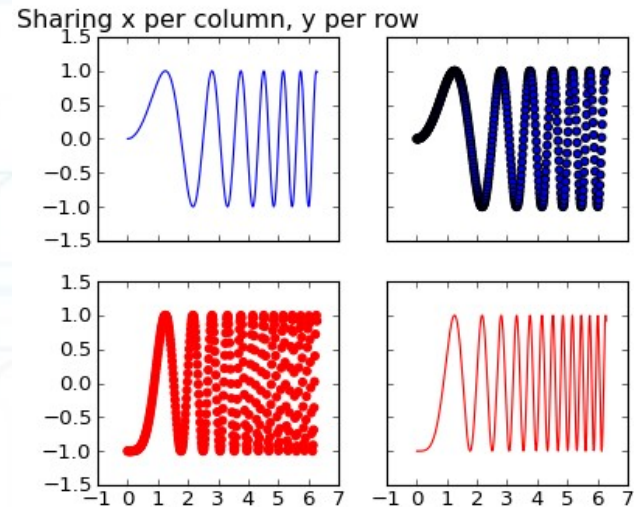
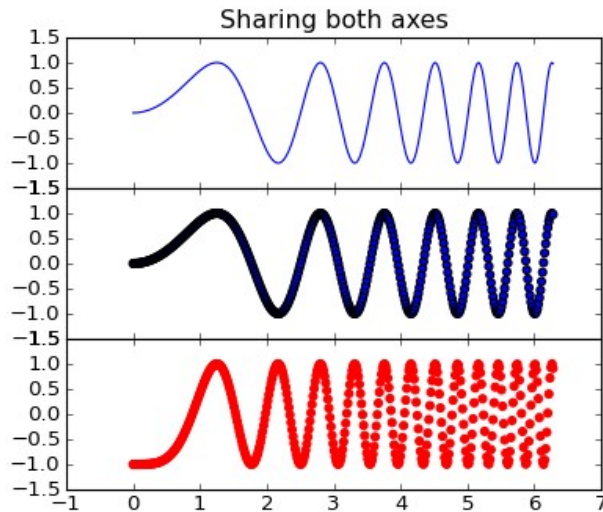
- 2 SUBPLOT
- Axes of subplots **unpacked**
- Sharing **y-axis**





# Example#2 Subplot

Need more?



# Example#3 Legend

```
import numpy as np
import matplotlib.pyplot as plt
```

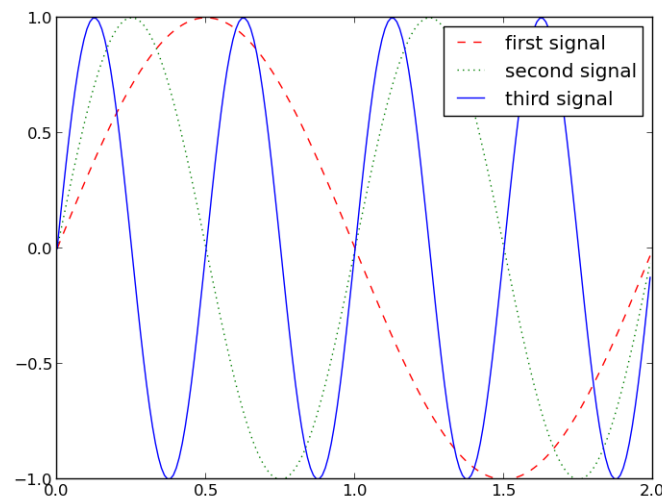
```
x = np.arange(0,2,.01)
y1 = np.sin(np.pi*x)
y2 = np.sin(2*np.pi*x)
y3 = np.sin(4*np.pi*x)
```

New way  
for Subplots

```
ax1 = plt.subplot(211)
ax2 = plt.subplot(212)
```

```
ax = plt.subplot(111)
plt.plot(x,y1,'r--',x,y2,'g:',x,y3,'b')
plt.legend(('first signal',
           'second signal',
           'third signal'))
```

```
plt.show()
```





# Modules

(<http://matplotlib.org/py-modindex.html> > 40)

here are some of them sound interesting or necessary to me:

- **matplotlib.figure**  
The figure module provides the top-level Artist, the Figure, which contains all the plot elements.
- **matplotlib.pyplot:**  
matlab like: combines pyplot with numpy
- **matplotlib.axes & .axis**
- **matplotlib.ticker**  
support completely configurable tick locating and formatting
- **matplotlib.lines**
- **matplotlib.cm (color map)**
- **matplotlib.colors**
- **matplotlib.legend**
- **matplotlib.gridspec**  
a module which specifies the location of the subplot in the figure
- **matplotlib.mathtext**
- **matplotlib.widgets**  
Widgets that are designed to work for any of the GUI backends
- **matplotlib.mlab:**  
Numerical python functions written for compatability with MATLAB commands with the same names.
- **matplotlib.projections & .scale**  
custom procedures that transform the data before it is displayed
- **matplotlib.dates**  
sophisticated date plotting capabilities, standing on the shoulders of python datetime
- **matplotlib.collections**  
Classes for the efficient drawing of large collections of objects that share most properties, e.g. a large number of line segments or polygons.
- **matplotlib.animation**
- **matplotlib.artist**  
<http://matplotlib.org/users/artists.html?highlight=artist%20how>



# Toolkits

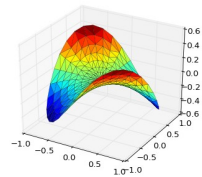
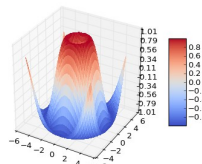
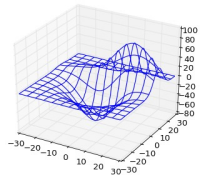
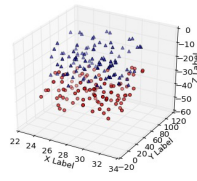
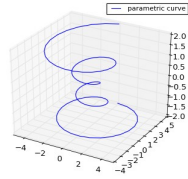
- **Basemap:**  
a library for plotting 2D data on maps in Python.
- **GTK Tools:**  
requires PyGTK (GTK+ for Python).
- **Excel Tools:**  
requires xlwt (Lib for spreadsheet files compatible with Excel).
- **Natgrid:**  
interface to natgrid C library for gridding irregularly spaced data.  
(natural neighbor interpolation method).
- **Mplot3d**  
some basic 3D plotting (scatter, surf, line, mesh).  
not the fastest or feature complete 3D library out there.
- **AxesGrid**  
ease displaying multiple images in matplotlib.

# mplot3d

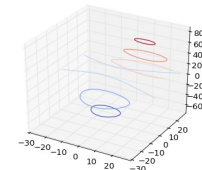
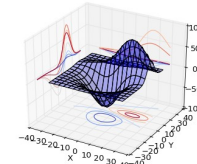
[http://matplotlib.org/mpl\\_toolkits/mplot3d/](http://matplotlib.org/mpl_toolkits/mplot3d/)

<http://matplotlib.org/examples/mplot3d/index.html#mplot3d-examples-index>

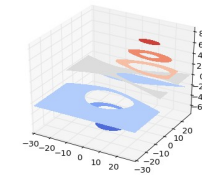
- Line
- Scatter
- Wireframe
- Surface
- Tri-Surface



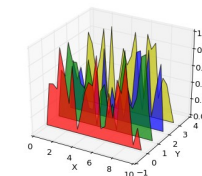
- Contour



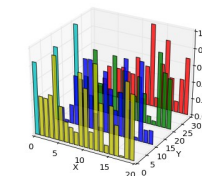
- Filled contour



- Polygon



- Bar





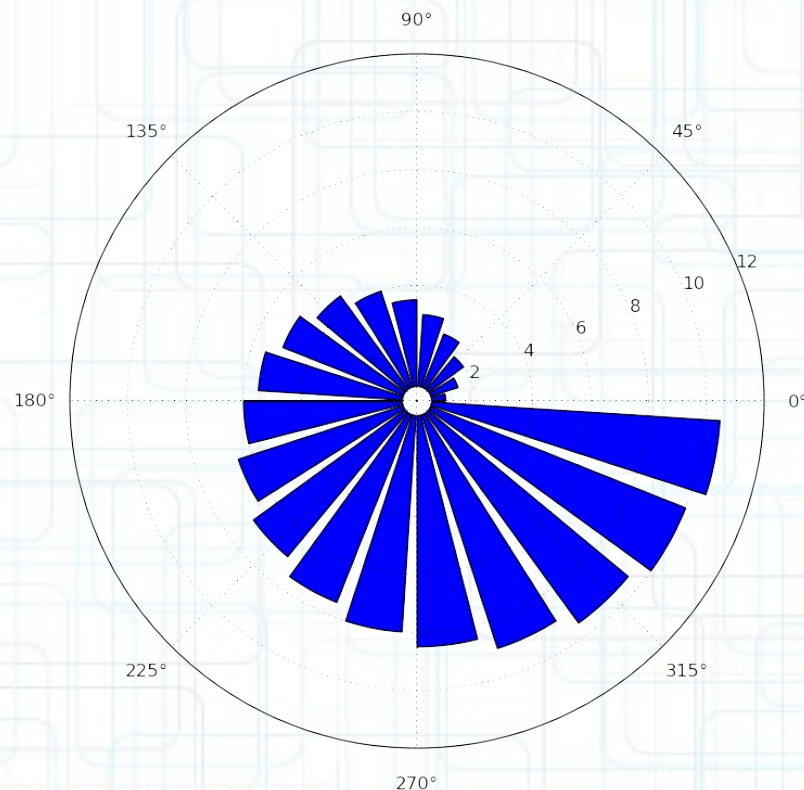
# Examples#4 – Polar Bar

```
import numpy as np
import matplotlib.cm as cm
from matplotlib.pyplot import figure, show, rc, savefig
```

```
fig = figure(figsize=(8,8))
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8], polar=True)
```

```
N = 20
theta = np.arange(0.0, 2*np.pi, 2*np.pi/N)
radii = np.arange(1,N+1)/2.0
width = np.ones(N)/4.0
ax.bar(theta, radii, width=width, bottom=0.5)
```

```
show()
```





# Examples#4 – Polar Bar

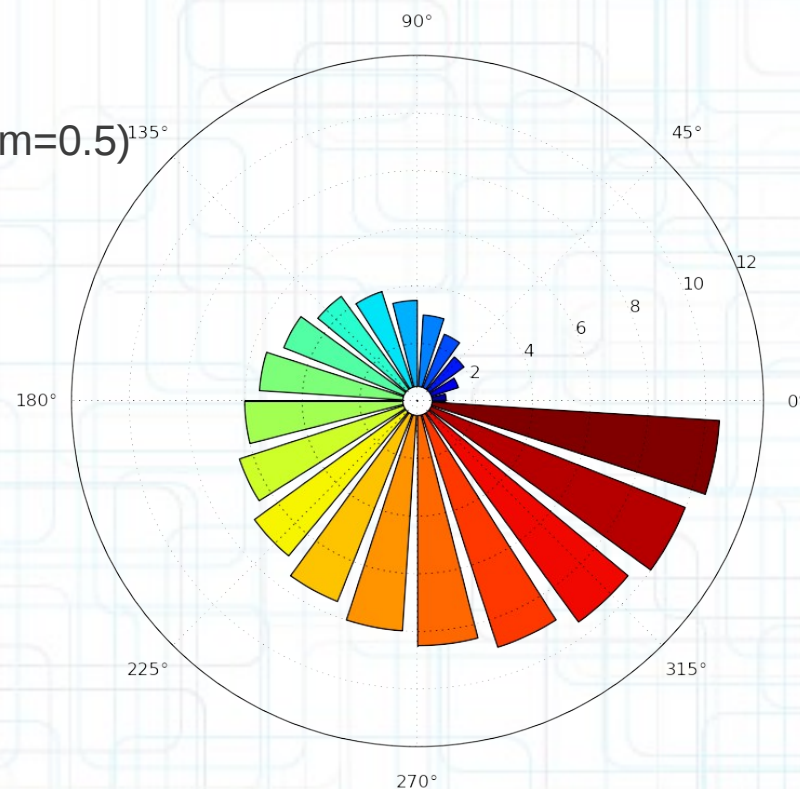
```
import numpy as np
import matplotlib.cm as cm
from matplotlib.pyplot import figure, show, rc, savefig
```

```
fig = figure(figsize=(8,8))
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8], polar=True)
```

```
N = 20
theta = np.arange(0.0, 2*np.pi, 2*np.pi/N)
radii = np.arange(1,N+1)/2.0
width = np.ones(N)/4.0
bars = ax.bar(theta, radii, width=width, bottom=0.5)
```

```
for r,bar in zip(radii, bars):
    bar.set_facecolor(cm.jet(r/10.))
    bar.set_alpha(0.5)
```

```
show()
```



# Examples#5 – Polar Scatter

```
from pylab import *
```

```
N = 30
```

```
theta = np.arange(0.0, 2*np.pi, 2*np.pi/N)
```

```
distance = arange(1,N+1)/2.0 # ones(N)
```

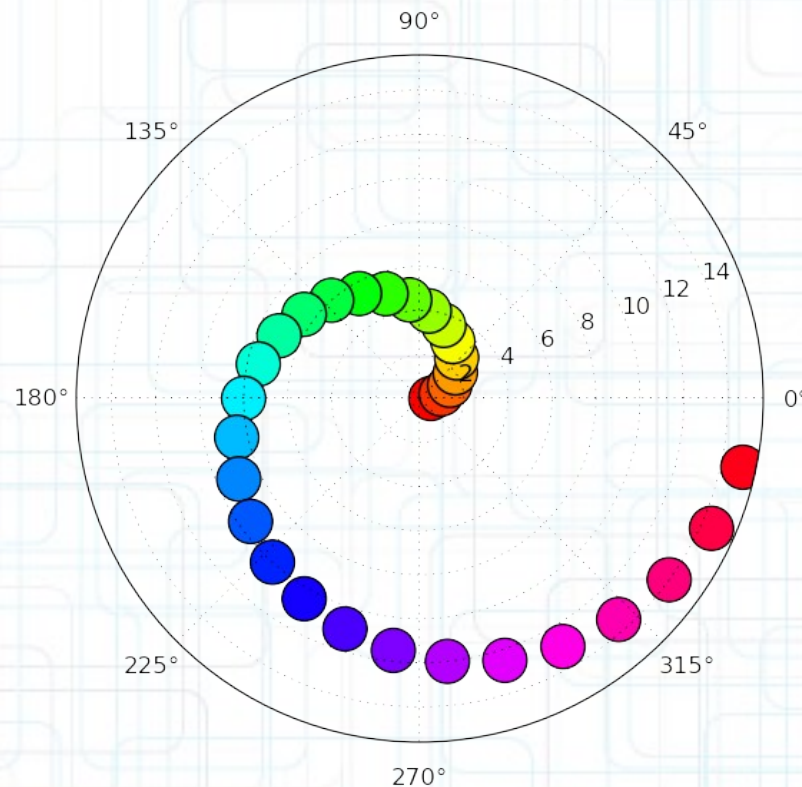
```
area = 500
```

```
colors = theta
```

```
ax = subplot(111, polar=True)
```

```
scatter(theta, distance,  
        c=colors, s=area,  
        cmap=cm.hsv)
```

```
show()
```





# Examples#6 - 3D Scatter

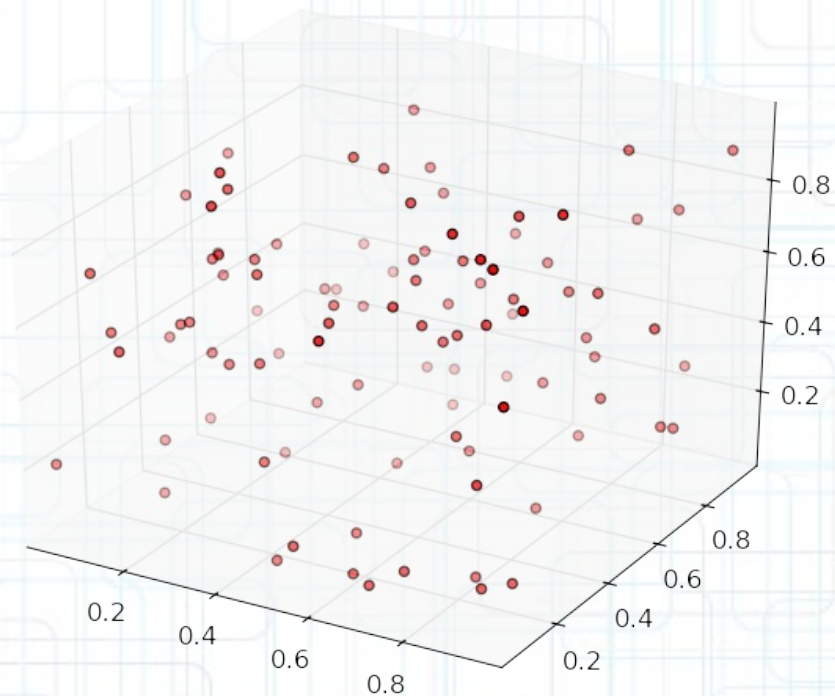
```
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
```

```
n = 100
```

```
xs = np.random.rand(n)
ys = np.random.rand(n)
zs = np.random.rand(n)
ax.scatter(xs, ys, zs, c='r', marker='o')
```

```
plt.show()
```





# Now What?

- Let's take an image (camera or file) and plot the distribution of pixels in HSV color space.

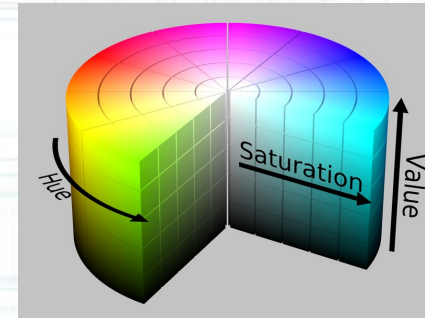


Image from wikipedia

- What do we need?
  - 2D plots in 3D (for HSV cylinder illustration)
  - Scatter 3D
  - Polar
  - Coloring
  - Sampling or averaging.